PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-161283
(43)Date of publication of application : 21.06.1996

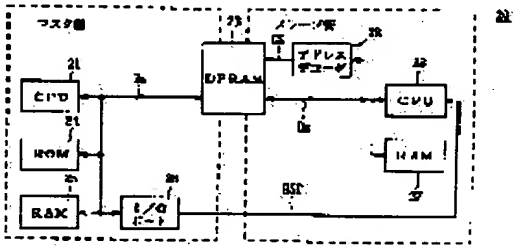
(51)Int.Cl. G06F 15/177

(21)Application number : 06-330694
(22)Date of filing : 07.12.1994

(71)Applicant : SONY CORP
(72)Inventor : KAMEYAMA NAOKI
SUGINO AKINOBU
YASUI HIROYUKI

(54) PLURAL PROCESSOR SYSTEMS

(57)Abstract:
PURPOSE: To embody plural processor systems whose constitutions can be further simplified.
CONSTITUTION: By making a slave processor 22 a reset state by a reset means 26, reading the boot code of the slave processor 22 from a memory 24, writing the code in the area corresponding to the boot code storage address of a shared memory 23, making the slave processor 22 read the boot code on the shared memory 23 by releasing the reset state of the slave processor 22 and starting the slave processor 22, the reading exclusive memory which has been required on a slave side to store the precode of a conventional slave processor 22 can be reduced. As a result, the constitution can be further simplified.



LEGAL STATUS

[Date of request for examination] 06.06.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号
特開平8－161283
(43)公開日　平成８年(1996)６月21日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/177				
			Ｇ０６Ｆ　15/ 16　４２０　Ｓ	

審査請求　未請求　請求項の数３　ＦＤ　（全 ９ 頁）

(21)出願番号　特願平6－330694

(22)出願日　平成６年(1994)12月７日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　亀山　直樹
東京都品川区北品川６丁目７番35号ソニー株式会社内
(72)発明者　杉野　彰信
東京都品川区北品川６丁目７番35号ソニー株式会社内
(72)発明者　安居　宏之
東京都品川区北品川６丁目７番35号ソニー株式会社内
(74)代理人　弁理士　田辺　恵基

(54)【発明の名称】　複数プロセツサシステム

(57)【要約】
【目的】本発明は複数プロセツサシステムに関し、一段と構成を簡易にし得る複数プロセツサシステムを実現する。
【構成】リセツト手段２６によつてスレーブプロセツサ２２をリセツト状態にし、メモリ２４からスレーブプロセツサ２２のブートコードを読み出して共有メモリ２３のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサ２２のリセツト状態を解除することによつてスレーブプロセツサ２２に共有メモリ２３上のブートコードを読み出させて当該スレーブプロセツサ２２を立ち上げることにより、従来スレーブプロセツサ２２のブートコードを格納しておくためスレーブ側に必要だつた読み出し専用メモリを削減することができ、これにより一段と構成を簡易にできる。

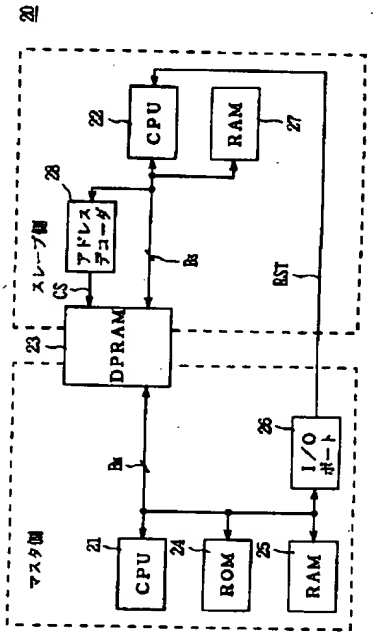


図1　複数CPUシステムの構成

1

【特許請求の範囲】

【請求項１】マスタプロセツサと少なくとも１つのスレーブプロセツサとを有し、当該マスタプロセツサとスレーブプロセツサとを共有メモリを介して接続してなる複数プロセツサシステムにおいて、

上記マスタプロセツサの制御に応じて上記スレーブプロセツサをリセツト状態にさせるリセツト手段と、

上記共有メモリを、上記スレーブプロセツサのブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダと、

上記スレーブプロセツサのブートコードが格納され、上記マスタプロセツサによつて読み出し可能なメモリとを具え、

上記リセツト手段によつて上記スレーブプロセツサをリセツト状態にし、上記マスタプロセツサによつて上記メモリから上記スレーブプロセツサのブートコードを読み出して上記共有メモリの上記ブートコード格納アドレスに対応する領域に書き込み、上記スレーブプロセツサのリセツト状態を解除することによつて上記スレーブプロセツサに上記共有メモリ上のブートコードを読み出させて当該スレーブプロセツサを立ち上げることを特徴とする複数プロセツサシステム。

【請求項２】マスタプロセツサと少なくとも１つのスレーブプロセツサとを有し、当該マスタプロセツサとスレーブプロセツサとを共有メモリを介して接続してなる複数プロセツサシステムにおいて、

上記マスタプロセツサの制御に応じて上記スレーブプロセツサをリセツト状態にさせるリセツト手段と、

上記共有メモリを、上記スレーブプロセツサのブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダと、

上記スレーブプロセツサによつて書き込み及び読み出し可能な第１のメモリと、

上記マスタプロセツサが上記共有メモリに書き込んだプログラムコードを読み出して上記第１のメモリに格納する命令内容でなる上記スレーブプロセツサのブートコード及び上記スレーブプロセツサのプログラムコードが格納され、上記マスタプロセツサによつて読み出し可能な第２のメモリと、

を具え、

上記リセツト手段によつて上記スレーブプロセツサをリセツト状態にし、上記マスタプロセツサによつて上記第２のメモリから上記スレーブプロセツサのブートコードを読み出して上記共有メモリの上記ブートコード格納アドレスに対応する領域に書き込み、上記スレーブプロセツサのリセツト状態を解除することによつて上記スレーブプロセツサに上記共有メモリ上のブートコードを読み出させて当該スレーブプロセツサを立ち上げ、さらに上記スレーブプロセツサに上記ブートコードの命令内容を実行させることによつて、上記マスタプロセツサが上記

2

第２のメモリから読み出して上記共用メモリ上に書き込んだプログラムコードを当該共用メモリから読み出して上記第１のメモリに格納することを特徴とする複数プロセツサシステム。

【請求項３】上記マスタプロセツサが上記第２のメモリから読み出して上記共用メモリ上に書き込んだプログラムコードを当該共用メモリから読み出して上記第１のメモリに格納する処理を繰り返し実行することを特徴とする請求項２に記載の複数プロセツサシステム。

【発明の詳細な説明】

【０００１】

【目次】以下の順序で本発明を説明する。

産業上の利用分野
従来の技術（図４）
発明が解決しようとする課題
課題を解決するための手段（図１～図３）
作用（図１～図３）
実施例（図１～図３）
発明の効果

【０００２】

【産業上の利用分野】本発明は複数プロセツサシステムに関し、例えば複数のＣＰＵ（Central Processing Unit）を有するシステムに適用して好適なものである。

【０００３】

【従来の技術】従来、複数のＣＰＵを有するシステム（以下これを複数ＣＰＵシステムと呼ぶ）においては、各ＣＰＵ間のインターフエースとして入出力ポートを２つ持ついわゆるデユアルポートＲＡＭ（Random Access Memory）が一般的に用いられている。

【０００４】例えば図４に示すように、複数ＣＰＵシステム１では、マスタ側のＣＰＵ２とスレーブ側のＣＰＵ３、４との間にそれぞれデユアルポートＲＡＭ（ＤＰＲＡＭ）５、６を設け、当該デユアルポートＲＡＭ５、６をマスタ側とスレーブ側の両方でアクセスすることによりマスタ側とスレーブ側との間でデータ等を受け渡すようになされている。ここで各ＣＰＵ２～４に対してはそれぞれ不揮発性メモリのＲＯＭ（Read Only Memory）７～９が設けられており、各ＣＰＵ２～４はそれぞれこのＲＯＭ７～９に格納されたプログラムコードに基づいて動作する。また各ＣＰＵ２～４に対してはそれぞれ揮発性メモリのＲＡＭ１０～１２が設けられており、各ＣＰＵ２～４はそれぞれこのＲＡＭ１０～１２に対して種々のデータを読み書きする。

【０００５】

【発明が解決しようとする課題】ところで複数ＣＰＵシステム１においては、スレーブ側のＣＰＵ３、４を動作させるプログラムコードの大部分をマスタ側のＲＯＭ７に格納しておき、そのプログラムコードをそれぞれデユアルポートＲＡＭ５、６を介してＲＡＭ１１、１２にコピーすることにより、ＲＯＭ８、９の容量を比較的小さ

くすることができると考えられる。

【０００６】しかしながら複数ＣＰＵシステム１では、上述のようにスレーブ側のＣＰＵ３、４を動作させるプログラムコードの大部分をマスタ側からコピーするようにしたとしても、ＣＰＵ３、４のブートコード（すなわちＣＰＵ３、４がリセット直後に読み出して立ち上がるために必要なコード）やＣＰＵ３、４がそれぞれデュアルポートＲＡＭ５、６を介してデータを送受信するときに必要なプログラムコードを格納するためにＲＯＭ８、９がどうしても必要である。すなわちＲＯＭ８、９の容量を小さくすることはできるが、比較的高価なＲＯＭ８、９を完全になくすことができず、この分全体として構成を簡易にし得ないと共に、コスト的に高くなる問題がある。

【０００７】またデュアルポートＲＡＭを用いて各ＣＰＵ間でデータ通信するものとしてこの他にも、特開平１－312659号公報に記載されるものがある。これに記載される複数ＣＰＵシステムでは、メインＣＰＵとトランスＣＰＵとの間でステータス情報を通信する場合、送信側はデュアルポートＲＡＭ上の所定のステータス情報を変更すると共に、変更したステータス情報を指示するヘツダをセツトし、受信側は変更されたステータス情報をヘツダによつて分析して当該変更されたステータス情報を読み出すようになされている。これによりこの複数ＣＰＵシステムでは、１回の通信で同時に複数のステータス情報を変更して送信でき、処理時間を低減することができる。しかしながらこの複数ＣＰＵシステムでも、通信する際に必要なプログラムコードを格納するためにＲＯＭを必要とし、この分全体として構成を簡易にし得ないと共に、コスト的に高くなる問題がある。

【０００８】本発明は以上の点を考慮してなされたもので、一段と構成を簡易にし得る複数プロセツサシステムを提案しようとするものである。

【０００９】

【課題を解決するための手段】かかる課題を解決するため本発明においては、マスタプロセツサ２１と少なくとも１つのスレーブプロセツサ２２とを有し、当該マスタプロセツサ２１とスレーブプロセツサ２２とを共有メモリ２３を介して接続してなる複数プロセツサシステム２０において、マスタプロセツサ２１の制御に応じてスレーブプロセツサ２２をリセツト状態にさせるリセツト手段２６と、共有メモリ２３を、スレーブプロセツサ２２のブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダ２８と、スレーブプロセツサ２２のブートコードが格納され、マスタプロセツサ２１によつて読み出し可能なメモリ２４とを設け、リセツト手段２６によつてスレーブプロセツサ２２をリセツト状態にし、マスタプロセツサ２１によつてメモリ２４からスレーブプロセツサ２２のブートコードを読み出して共有メモリ２３のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサ２２のリセツト状態を解除することによつてスレーブプロセツサ２２に共有メモリ２３上のブートコードを読み出させて当該スレーブプロセツサ２２を立ち上げるようにした。

【００１０】また本発明においては、マスタプロセツサ２１と少なくとも１つのスレーブプロセツサ２２とを有し、当該マスタプロセツサ２１とスレーブプロセツサ２２とを共有メモリ２３を介して接続してなる複数プロセツサシステム２０において、マスタプロセツサ２１の制御に応じてスレーブプロセツサ２２をリセツト状態にさせるリセツト手段２６と、共有メモリ２３を、スレーブプロセツサ２２のブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダ２８と、スレーブプロセツサ２２によつて書き込み及び読み出し可能な第１のメモリ２７と、マスタプロセツサ２１が共有メモリ２３に書き込んだプログラムコードを読み出して第１のメモリ２７に格納する命令内容でなるスレーブプロセツサ２２のブートコード及びスレーブプロセツサ２２のプログラムコードが格納され、マスタプロセツサ２１によつて読み出し可能な第２のメモリ２４とを設け、リセツト手段２６によつてスレーブプロセツサ２２をリセツト状態にし、マスタプロセツサ２１によつて第２のメモリ２４からスレーブプロセツサ２２のブートコードを読み出して共有メモリ２３のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサ２２のリセツト状態を解除することによつてスレーブプロセツサ２２に共有メモリ２３上のブートコードを読み出させて当該スレーブプロセツサ２２を立ち上げ、さらにスレーブプロセツサ２２にブートコードの命令内容を実行させることによつて、マスタプロセツサ２１が第２のメモリ２４から読み出して共用メモリ２３上に書き込んだプログラムコードを当該共用メモリ２３から読み出して第１のメモリ２７に格納するようにした。

【００１１】また本発明においては、マスタプロセツサ２１が第２のメモリ２４から読み出して共用メモリ２３上に書き込んだプログラムコードを当該共用メモリ２３から読み出して第１のメモリ２７に格納する処理を繰り返し実行するようにした。

【００１２】

【作用】リセツト手段２６によつてスレーブプロセツサ２２をリセツト状態にし、マスタプロセツサ２１によつてメモリ２４からスレーブプロセツサ２２のブートコードを読み出して共有メモリ２３のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサ２２のリセツト状態を解除することによつてスレーブプロセツサ２２に共有メモリ２３上のブートコードを読み出させて当該スレーブプロセツサ２２を立ち上げるようにしたことにより、従来スレーブプロセツサ２２のブートコードを格納しておくためスレーブ側に必要だつた読み出し専用メモリを削減することができる。

5

【００１３】またリセット手段２６によつてスレーブプロセツサ２２をリセット状態にし、マスタプロセツサ２１によつて第２のメモリ２４からスレーブプロセツサ２２のブートコードを読み出して共有メモリ２３のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサ２２のリセット状態を解除することによつてスレーブプロセツサ２２に共有メモリ２３上のブートコードを読み出させて当該スレーブプロセツサ２２を立ち上げ、さらにスレーブプロセツサ２２にブートコードの命令内容を実行させることによつて、マスタプロセツサ２１が第２のメモリ２４から読み出して共用メモリ２３上に書き込んだプログラムコードを当該共用メモリ２３から読み出して第１のメモリ２７に格納するようにしたことにより、第２のメモリ２４に格納されたスレーブプロセツサ２２のプログラムコードを第１のメモリ２７にコピーできると共に、従来スレーブプロセツサ２２のブートコードやプログラムコードを格納しておくためスレーブ側に必要だつた読み出し専用メモリを削減することができる。

【００１４】またマスタプロセツサ２１が第２のメモリ２４から読み出して共用メモリ２３上に書き込んだプログラムコードを当該共用メモリ２３から読み出して第１のメモリ２７に格納する処理を繰り返し実行するようにしたことにより、共用メモリ２３の容量よりも大きなプログラムコードを第１のメモリ２７にコピーできる。

【００１５】

【実施例】以下図面について、本発明の一実施例を詳述する。

【００１６】図１において、２０は全体として本発明を適用した複数ＣＰＵシステムを示し、マスタ側のＣＰＵ２１とスレーブ側のＣＰＵ２２との間にデユアルポートＲＡＭ（ＤＰＲＡＭ）２３を設け、当該デユアルポートＲＡＭ２３をマスタ側とスレーブ側の両方でアクセスすることによりマスタ側とスレーブ側の間で種々のコードやデータを受け渡すようになされている。

【００１７】この実施例の場合、マスタ側には不揮発性メモリのＲＯＭ２４が設けられ、このＲＯＭ２４にマスタ側のＣＰＵ２１を動作させるプログラムコードやスレーブ側のＣＰＵ２２を動作させるプログラムコード（リセット解除後にＣＰＵ２２が読み出して実行するブートコードを含む）が格納されている。またマスタ側には揮発性メモリのＲＡＭ２５が設けられており、マスタ側のＣＰＵ２５はこのＲＡＭ２５を作業領域として種々のデータを読み書きする。さらにマスタ側にはＣＰＵ２１の指示に応じてリセット信号ＲＳＴを生成する入出力ポートいわゆるＩ／Ｏポート２６が設けられており、このリセット信号ＲＳＴをスレーブ側のＣＰＵ２２に対して出力して当該ＣＰＵ２２をリセット状態にするようになされている。この場合、ＣＰＵ２１、デユアルポートＲＡＭ２３、ＲＯＭ２４、ＲＡＭ２５及びＩ／Ｏポート２６はそれぞれバスＢ$_M$によつて接続され、このバスＢ$_M$を介して書き込み制御信号、読み出し制御信号、セレクト信号、アドレス信号、データ等の種々の情報を受け渡すようになされている。

6

【００１８】これに対してスレーブ側には揮発性メモリのＲＡＭ２７が設けられており、デユアルポートＲＡＭ２３を介してコピーしたＣＰＵ２２を動作させるためのプログラムコードをこのＲＡＭ２７に格納するようになされている。またＲＡＭ２７はＣＰＵ２２の作業領域としても使用され、種々のデータが読み書きされる。さらにスレーブ側には簡単なＰＬＤ（Programmable Logic Device）で構成されたアドレスデコーダ２８が設けられており、このアドレスデコーダ２８によつてＣＰＵ２２のアドレス信号をデコードし、その結果得た信号をチツプセレクト信号ＣＳとしてデユアルポートＲＡＭ２３に供給することにより、当該デユアルポートＲＡＭ２３をブートコード格納アドレスを含む領域に割り当てるようになされている。この場合、ＣＰＵ２２、ＲＡＭ２７、デユアルポートＲＡＭ２３及びアドレスデコーダ２８はそれぞれバスＢ$_S$によつて接続され、このバスＢ$_S$を介して書き込み制御信号、読み出し制御信号、セレクト信号、アドレス信号、データ等の種々の情報を受け渡すようになされている。

【００１９】ここでデユアルポートＲＡＭ２３を例えば16〔Kbit〕のもの２つで構成し、ＲＡＭ２７を例えば256〔Kbit〕のスタテイツクＲＡＭ（ＳＲＡＭ）４つで構成した場合には、スレーブ側のＣＰＵ２２のメモリマツプを例えば図２（Ａ）に示すように構成する。この図２（Ａ）に示すように、スレーブ側のＣＰＵ２２のメモリ領域（00000(h)～FFFFF(h)番地）に対して、リセット直後にＣＰＵ２２が読み出すアドレスはFFFF0(h)番地であるため、デユアルポートＲＡＭのアドレスをアドレスデコーダ２８によつてFFFF0(h)番地を含む領域（すなわちF0000(h)～FFFFF(h)番地の領域）に割り当てる。

【００２０】因みに、図２（Ａ）においては、F0000(h)～FFFFF(h)番地の領域（すなわち64〔KByte〕）をデユアルポートＲＡＭ領域としているが、使用するデユアルポートＲＡＭが16〔Kbit〕のもの２つであるため（すなわち４〔KByte〕であるため）、実際には、図２（Ｂ）に示すように、１６個（F?000(h)～F?FFF(h)番地；?=0～F）の４〔KByte〕の領域に対して２つのデユアルポートＲＡＭを繰り返し割り当てる。またマスタ側のＣＰＵ２１から見たデユアルポートＲＡＭのアドレスをC0000(h)～COFFF(h)番地にすれば、マスタ側のアドレスC0XXX(h)番地とスレーブ側のアドレスF?XXX(h)番地（? ＝0～F）が対応するアドレスになる。従つてスレーブ側のＣＰＵ２２のブートコードはマスタ側からCOFF0(h)番地に書き込まれる。

【００２１】ここでこのような構成を有する複数ＣＰＵシステム２０においては、図３に示すような処理手順を

7　　8

実行することにより、マスタ側のＣＰＵ２１からスレーブ側のＣＰＵ２２を立ち上げると共に、マスタ側からスレーブ側にプログラムコードをダウンロード（すなわち転送）する。まずマスタ側のＣＰＵ２１は、電源投入後ステップＳＰ１から入つて続くステップＳＰ２において、Ｉ／Ｏポート２６を制御してスレーブ側のＣＰＵ２２に対してリセット信号ＲＳＴを出力することにより、当該スレーブ側のＣＰＵ２２をリセット状態にさせる。次にマスタ側のＣＰＵ２１は、ステップＳＰ３において、ＲＯＭ２４からスレーブ側のＣＰＵ２２のブートコードを読み出し、デュアルポートＲＡＭ２３に書き込む。

【００２２】この場合、ＣＰＵ２２のブートコードはリセット解除直後に当該ＣＰＵ２２が読みに行く領域に書き込まれる（すなわち上述のメモリマップ例によれば、マスタ側から見てCOFFO(h)番地の領域）。またこのとき書き込まれるブートコードには、スレーブ側のＣＰＵ２２がデュアルポートＲＡＭ２３に書き込まれているコードをＲＡＭ２７にコピーすると共に、コピー終了後にＣＰＵ２２がそのコピーしたコードに基づいて動作するような情報が含まれている。

【００２３】次にマスタ側のＣＰＵ２１は、ステップＳＰ４においてＩ／Ｏポート２６を制御してリセット信号ＲＳＴを解除し、スレーブ側のＣＰＵ２２のリセット状態を解除する。そしてマスタ側のＣＰＵ２１はステップＳＰ５に移り、スレーブ側からの追加コード転送要求を待つ。この状態において、追加コード転送要求が発生すると、マスタ側のＣＰＵ２１はステップＳＰ６に移り、ＲＯＭ２４から追加コードを読み出して当該追加コードをデュアルポートＲＡＭ２３に書き込むと共に、スレーブ側のＣＰＵ２２に対して読み出し要求を出力する。この読み出し要求の出力はデュアルポートＲＡＭ２３が持つ割り込み発生機能によつて実現される。因みに、この実施例の場合には、デュアルポートＲＡＭ２３上にフラグを用意しておき、そのフラグを判別することによつて割り込みが読み出し要求であるかそれ以外のものであるかを区別するようになされている。

【００２４】次にマスタ側のＣＰＵ２１は、ステップＳＰ７においてスレーブ側に転送すべきコードの有無を判断し、転送すべきコードがある場合には再びステップＳＰ５に戻つて同様の処理を繰り返し、転送すべきコードがない場合（すなわち転送すべきコードを全て転送し終えた場合）にはステップＳＰ８に移つて転送手順を終了する。因みに、転送手順を終了する場合、最後に転送するコードの後ろに所定のデータを付加することによつて転送手順の終了をスレーブ側のＣＰＵ２２に対して通知する。

【００２５】このようなマスタ側のＣＰＵ２１の処理に対して、スレーブ側のＣＰＵ２２は電源投入後ステップＳＰ１０から入り、リセット信号ＲＳＴの受信によつて続くステップＳＰ１１でリセット状態になり、動作を停止する。そしてスレーブ側のＣＰＵ２２はリセット信号ＲＳＴが解除されると続くステップＳＰ１２でリセット状態を解除する。そしてスレーブ側のＣＰＵ２２はステップＳＰ１３においてデュアルポートＲＡＭ２３からブートコードを読み出し（すなわち上述のメモリマップ例によれば、スレーブ側から見てFFFF0(h)番地の領域を読みに行く）、そのブートコードをプログラムとして動作を開始して立ち上がる。この場合、ブートコードには上述したようにスレーブ側のＣＰＵ２２がデュアルポートＲＡＭ２３に書き込まれているコードをＲＡＭ２７にコピーすると共に、コピー終了後ＣＰＵ２２がそのコピーしたコードに基づいて動作するような情報が含まれている。

【００２６】このためスレーブ側のＣＰＵ２２は、続くステップＳＰ１４においてデュアルポートＲＡＭ２３上のコードをＲＡＭ２７にコピーし、そのＲＡＭ２７上のコードに基づいて動作する（すなわち動作形態をＲＡＭ２７上に移す）。次にスレーブ側のＣＰＵ２２はステップＳＰ１５で追加コード転送要求を転送要求待ち状態にあるマスタ側のＣＰＵ２１に対して出力した後、マスタ側からの読み出し要求を待つ。この場合、転送要求の出力はデュアルポートＲＡＭ２３が持つ割り込み発生機能によつて実現される。因みに、この実施例の場合には、デュアルポートＲＡＭ２３上にフラグを用意しておき、そのフラグを判別することによつて割り込みが転送要求であるかそれ以外のものであるかを区別するようになされている。

【００２７】スレーブ側のＣＰＵ２２はマスタ側から読み出し要求が発生すると、続くステップＳＰ１６に移り、デュアルポートＲＡＭ２３上のコードをＲＡＭ２７にコピーし、そのＲＡＭ２７上のコードに基づいて動作する。次にスレーブ側のＣＰＵ２２はステップＳＰ１７において受信するコードの有無を判断し、受信するコードがまだある場合には再びステップＳＰ１５に戻つて同様の処理を繰り返し、受信するコードがない場合にはステップＳＰ１８に移つて処理を終了する。因みに、受信するコードの有無を判断する場合、ＣＰＵ２２は転送されて来たコードに転送終了を意味する所定のコードが或るか否かを見て判断する。

【００２８】このような処理手順をマスタ側のＣＰＵ２１とスレーブ側のＣＰＵ２２とがそれぞれ実行することにより、ＲＯＭ２４に格納されているスレーブ側のＣＰＵ２２を動作させるプログラムコードを転送することができる。因みに、マスタ側のＣＰＵ２１、スレーブ側のＣＰＵ２２がそれぞれ動作しているときにマスタ側とスレーブ側との間でデータを転送する場合には、上述のようにリセット信号ＲＳＴを用いず、デュアルポートＲＡＭ２３の割り込み発生機能のみよつて行われる。すなわち送信側がデュアルポートＲＡＭ２３にデータを書き込

9

んだときに受信側に対して割り込み要求を出力し、受信側はこの割り込み要求に応じてデユアルポートＲＡＭ２３に書き込まれているデータを読み出す。これにより各ＣＰＵ２１、２２の動作を止めることなく、各ＣＰＵ２１、２２間でデータを転送することができる。

【００２９】以上の構成において、電源投入後、まずマスタ側のＣＰＵ２１はＩ／Ｏポート２６によつてスレーブ側のＣＰＵ２２に対してリセツト信号ＲＳＴを出力し、当該ＣＰＵ２２をリセツト状態にさせる。そしてマスタ側のＣＰＵ２１は、ＣＰＵ２２のリセツト状態を維持したままＲＯＭ２４からＣＰＵ２２のブートコードを読み出してデユアルポートＲＡＭ２３に書き込む。この場合、ブートコードはリセツト解除後にＣＰＵ２２が読みに行くデユアルポートＲＡＭ２３の領域に対して書き込まれる。ブートコードの書き込みが終了すると、マスタ側のＣＰＵ２１はリセツト信号ＲＳＴを解除してスレーブ側のＣＰＵ２２のリセツト状態を解除すると共に、追加コードの転送要求待ち状態に入る。

【００３０】一方、スレーブ側のＣＰＵ２２はリセツト信号ＲＳＴの解除によつて動作を開始し、デユアルポートＲＡＭ２３の所定の領域（すなわちアドレスデコーダ２８によつて設定されたブートコードが書き込まれている領域）を読み出してそれを実行する。これによりスレーブ側のＣＰＵ２２が立ち上がる。この場合、ブートコードにはデユアルポートＲＡＭ２３上のコードをＲＡＭ２７にコピーし、コピー終了後そのコピーしたコード上に自身の動作を移す手順が記されているため、ＣＰＵ２２はこの指示に従つてデユアルポートＲＡＭ２３上のコードをＲＡＭ２７にコピーすると共に、コピー終了後ＲＡＭ２７にコピーしたコード上に自身の動作を移す。次にＣＰＵ２２は追加コード転送要求をデユアルポートＲＡＭ２３を介して転送要求待ち状態にあるマスタ側のＣＰＵ２１に対して送出すると共に、読み出し要求待ち状態に入る。

【００３１】マスタ側のＣＰＵ２１は、この追加コード転送要求に応じてＲＯＭ２４から追加すべきコードを読み出してデユアルポートＲＡＭ２３に書き込むと共に、読み出し要求をデユアルポートＲＡＭ２３を介してスレーブ側のＣＰＵ２２に対して送出する。スレーブ側のＣＰＵ２２は、この読み出し要求に応じてデユアルポートＲＡＭ２３上のコードをコピーし、そのコピーしたコード上に自身の動作を移す。そしてＣＰＵ２２は、追加するコードがまだある場合には、再び追加コード転送要求を送出し、追加するコードがなくなるまでマスタ側のＣＰＵ２１との間で転送手順を繰り返す。そしてスレーブ側のＣＰＵ２２は追加するコードがなくなつたら転送手順を終了し、ＲＡＭ２７上のプログラムコードを実行する。

【００３２】このようにして複数ＣＰＵシステム２０では、デユアルポートＲＡＭ２３を介してスレーブ側のＣＰＵ２２を立ち上げると共に、デユアルポートＲＡＭ２３よりも容量が大きいＣＰＵ２２のプログラムコードをスレーブ側に転送することができる。これにより従来必要であつた比較的高価なＲＯＭをスレーブ側から削除することができ、全体として構成を簡易にできると共に、コストダウンすることができる。また複数ＣＰＵシステム２０では、スレーブ側のＣＰＵ２２のプログラムコードをマスタ側のＲＯＭ２４に書き込んでおき、それをスレーブ側に転送するため、特にスレーブ側のＣＰＵ２２が増えた場合には、システムのソフトウエア管理を一元化できると共に、ソフトウエアのバージヨンアツプ時に煩雑なＲＯＭ交換作業を減らすことができる。

10

【００３３】以上の構成によれば、マスタ側のＣＰＵ２１からスレーブ側のＣＰＵ２２をリセツト状態にするＩ／Ｏポート２６と、スレーブ側のＣＰＵ２２から見たデユアルポートＲＡＭ２３のアドレスをブートコード格納アドレスを含む領域に設定するアドレスデコーダ２８とを設け、マスタ側のＣＰＵ２１からスレーブ側のＣＰＵ２２をリセツト状態にしてデユアルポートＲＡＭ２３にＣＰＵ２２のブートコードを書き込んだ後、ＣＰＵ２２のリセツト状態を解除することにより、スレーブ側のＣＰＵ２２にブートコードを読み出させて当該スレーブ側のＣＰＵ２２を立ち上げることができる。これにより従来必要だつたＲＯＭを削除することができ、全体として構成を簡易にできる。

【００３４】またブートコード中に、マスタ側からデユアルポートＲＡＭ２３に書き込んだコードをＲＡＭ２７にコピーする命令を含ませておくことにより、スレーブ側のＣＰＵ２２を立ち上げた後、ＣＰＵ２２を動作させるプログラムコードをマスタ側からスレーブ側に転送することができる。

【００３５】なお上述の実施例においては、プロセツサとしてＣＰＵ２１、２２が用いられたシステムについて述べたが、本発明はこれに限らず、プロセツサとしてＤＳＰ（Digital Signal Processor）が用いられたシステムでも良く、要はプロセツサを複数用いたシステムであれば本発明を適用し得る。

【００３６】また上述の実施例においては、アドレスデコーダ２８をＰＬＤによつて構成した場合について述べたが、本発明はこれに限らず、汎用ロジツクを用いて構成しても良い。

【００３７】さらに上述の実施例においては、デユアルポートＲＡＭ２３を用いてＲＯＭ２４上のコードをスレーブ側のＲＡＭ２７に転送した場合について述べたが、本発明はこれに限らず、転送に際してＤＭＡ（Direct Memory Access）を用いても良い。

【００３８】また上述の実施例においては、デユアルポートＲＡＭ２３のアドレスをF0000(h)～FFFFF(h)番地の領域に割り当てた場合について述べたが、本発明はこれに限らず、リセツト直後にＣＰＵ２２が読み出すアドレ

11　　12

スを含む領域に設定しさえすれば、デユアルポートＲＡＭ２３のアドレスとしては他の領域でも良い。

【００３９】さらに上述の実施例においては、マスタ側からスレーブ側にコードを転送する際（図３参照）、転送手順の終了をマスタ側からスレーブ側に通知する場合について述べたが、本発明はこれに限らず、スレーブ側からマスタ側に通知するようにしても良い。

【００４０】また上述の実施例においては、不揮発性メモリとしてＲＯＭ２４を用いた場合について述べたが、本発明はこれに限らず、不揮発性メモリとしてフラツシユメモリ等を用いても良い。

【００４１】さらに上述の実施例においては、Ｉ／Ｏポート２６によつてリセツト信号ＲＳＴを生成する場合について述べたが、本発明はこれに限らず、ＣＰＵ２１が出力ポートを有するものであれば当該ＣＰＵ２１でリセツト信号ＲＳＴを生成するようにしても良い。

【００４２】

【発明の効果】上述のように本発明によれば、リセツト手段によつてスレーブプロセツサをリセツト状態にし、マスタプロセツサによつてメモリからスレーブプロセツサのブートコードを読み出して共有メモリのブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサのリセツト状態を解除することによつてスレーブプロセツサに共有メモリ上のブートコードを読み出させて当該スレーブプロセツサを立ち上げるようにしたことにより、従来スレーブプロセツサのブートコードを格納しておくためスレーブ側に必要だつた読み出し専用メモリを削減することができる。かくするにつき一段と構成を簡易にし得る複数プロセツサシステムを実現できる。

【００４３】またリセツト手段によつてスレーブプロセツサをリセツト状態にし、マスタプロセツサによつて第２のメモリからスレーブプロセツサのブートコードを読み出して共有メモリのブートコード格納アドレスに対応する領域に書き込み、スレーブプロセツサのリセツト状態を解除することによつてスレーブプロセツサに共有メモリ上のブートコードを読み出させて当該スレーブプロセツサを立ち上げ、さらにスレーブプロセツサにブートコードの命令内容を実行させることによつて、マスタプロセツサが第２のメモリから読み出して共用メモリ上に書き込んだプログラムコードを当該共用メモリから読み出して第１のメモリに格納するようにしたことにより、第２のメモリに格納されたスレーブプロセツサのプログラムコードを第１のメモリにコピーできると共に、従来スレーブプロセツサのブートコードやプログラムコードを格納しておくためスレーブ側に必要だつた読み出し専用メモリを削減することができる。かくするにつき一段と構成を簡易にし得る複数プロセツサシステムを実現できる。

【図面の簡単な説明】

【図１】本発明の一実施例による複数ＣＰＵシステムの構成を示すブロツク図である。

【図２】スレーブ側のＣＰＵのメモリマツプを示す略線図である。

【図３】スレーブ側にプログラムコードを転送する際の手順を示すフローチヤートである。

【図４】従来の複数ＣＰＵシステムの構成を示すブロツク図である。

【符号の説明】

１、２０……複数ＣＰＵシステム、２、２１……マスタ側のＣＰＵ、３、４、２２……スレーブ側のＣＰＵ、５、６、２３……デユアルポートＲＡＭ、７～９、２４……ＲＯＭ、１０～１２、２５、２７……ＲＡＭ、２６……Ｉ／Ｏポート、２８……アドレスデコーダ。

【図１】

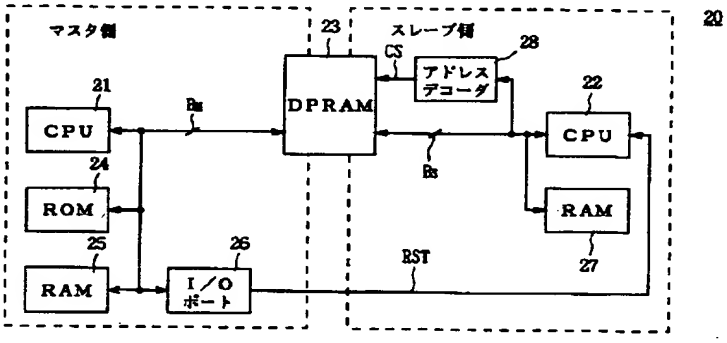


図１　複数ＣＰＵシステムの構成

【図２】

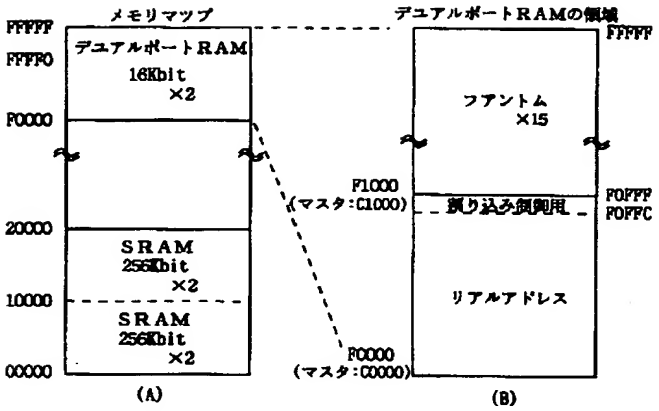


図２　スレーブ側のＣＰＵのメモリマップ

【図４】

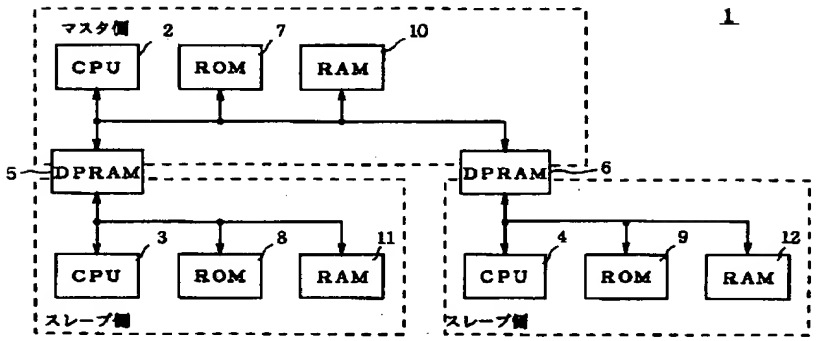


図４　従来の複数ＣＰＵシステムの構成

【図３】

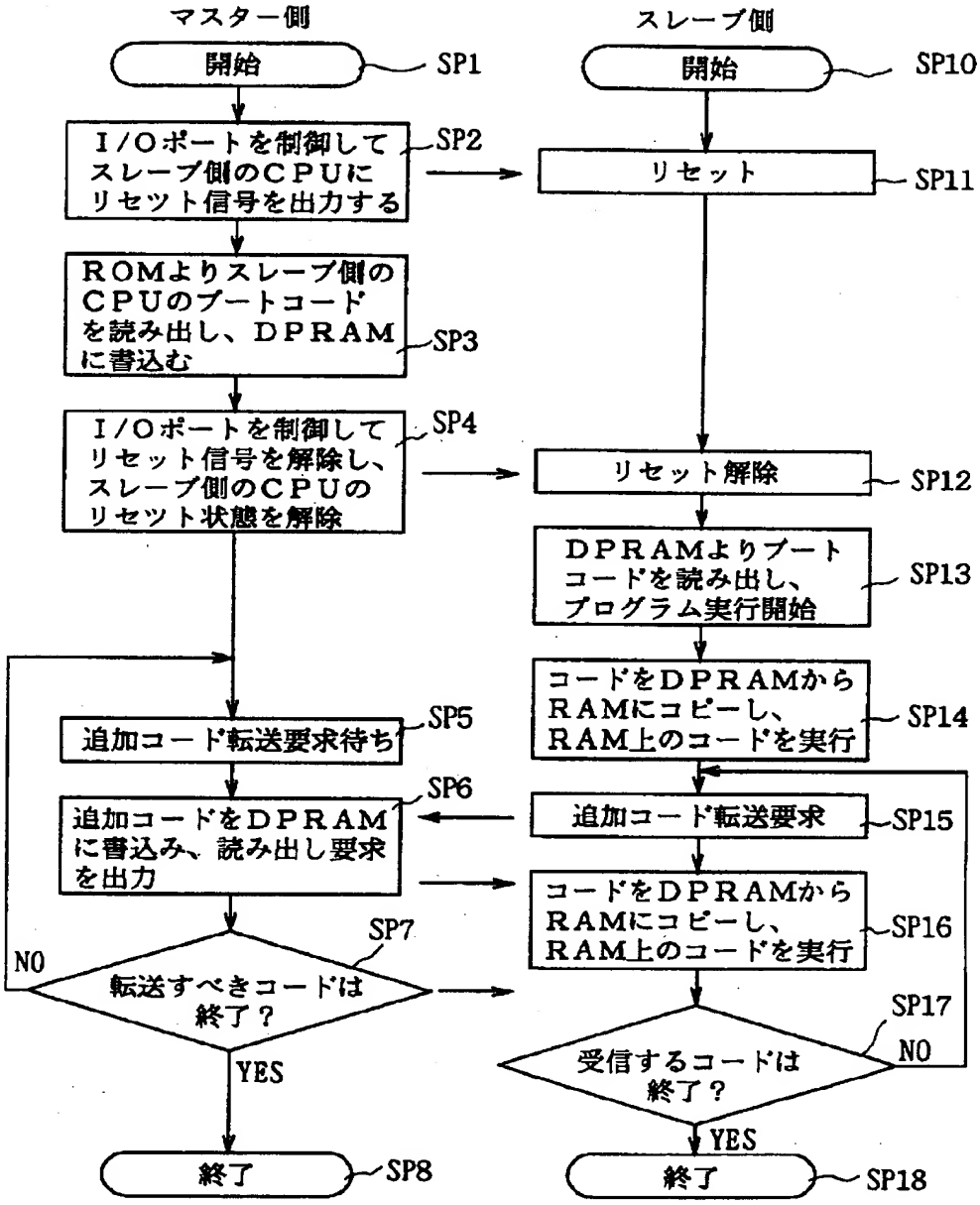


図３　スレーブ側にプログラムコードを転送する手順